Accurate simulations of planar topological codes cannot use cyclic boundaries

Austin G. Fowler
Centre for Quantum Computation and Communication Technology, School of Physics, The University of Melbourne, Victoria 3010, Australia
(Dated: September 18, 2012)

Cyclic boundaries are used in many branches of physics and mathematics, typically to assist the approximation of a large space. We show that when determining the performance of planar, fault-tolerant, topological quantum error correction, using cyclic boundaries leads to a significant underestimate of the logical error rate. We present cyclic and non-cyclic surface code simulations exhibiting this discrepancy, and analytic formulae precisely reproducing and explaining the observed behavior in the limit of low physical error.

Topological quantum error correction (TQEC) is the lowest overhead technique for achieving reliable large-scale quantum computation given a 2-D lattice of qubits with nearest neighbor interactions [1]. As such, accurate simulations of the performance of such codes under physically realistic assumptions are of great practical relevance. Much of the existing literature on TQEC uses cyclic boundaries [2–5]. While this is sufficient to obtain the threshold error rate $p_{\rm th}$ of a given TQEC scheme, we show that this is not sufficient to obtain an accurate logical error rate at a given code distance d and physical error rate $p < p_{\rm th}$. Indeed, we show that cyclic boundaries lead to an exponentially growing underestimate of the logical error rate with d.

Without loss of generality, we shall focus on the surface code [6–10] making use of minimum weight perfect matching for decoding [11–14]. Our results are relevant to any periodic, fault-tolerant implementation of any planar TQEC code. An up-to-date and detailed review of the surface code can be found in [15]. We shall use the error models (standard depolarizing) and circuits (8-step error detection cycle using only Z basis initialization and measurement) described in this review.

The patterns of stabilizers [16] associated with distance $d = 3$ non-cyclic and cyclic surface codes are shown in Fig. 1. Consider Fig. 2. This shows how errors propagate and create detection events both away from and near boundaries of the qubit array (the latter only when assuming non-cyclic boundaries). It is conceptually straightforward to analyze the propagation of all possible single errors and determine the total probability of any given pair of detection events due to all distinct single errors. We can visualize each such probability as a cylinder with space-time endpoints corresponding to the detection events and diameter proportional to the total probability. We call such a visualization a nest, and each cylinder a stick. Nests for X stabilizer measurement in the surface code assuming non-cyclic and cyclic boundaries are shown in Figs. 3 and 4, respectively. These figures were generated using our tool Autotune [17], modified to perform a simple sum of probabilities rather than the more complex polynomial used in prior work [18]. Due to the complexity of these figures, their raw data has been included in the Supplementary Material.

The probability of each type of logical error per round of error detection for various distances d and non-cyclic and cyclic boundaries as a function of the physical depolarizing error rate p is shown in Figs. 5–10. It can be seen that the logical error rates of the cyclic case are significantly lower, particularly logical X_1 and Z_2 errors. We can determine the low p asymptotic forms of the curves

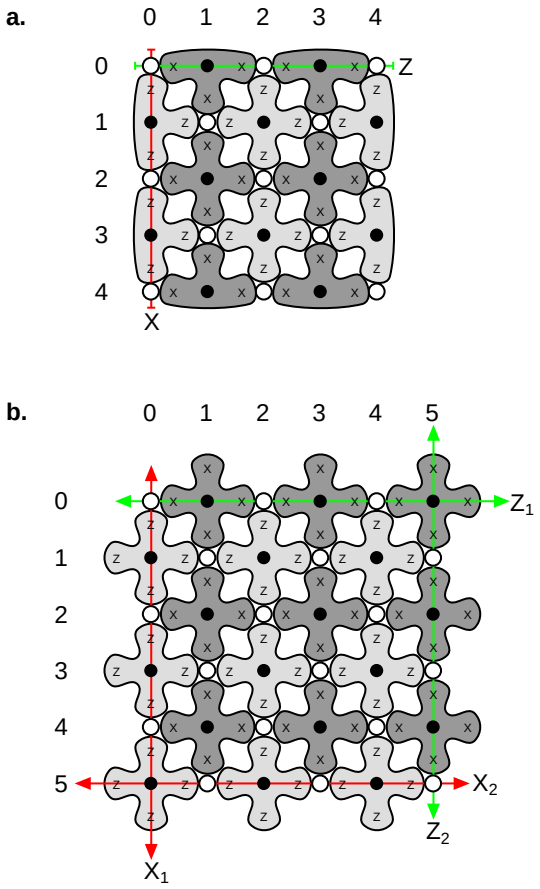

FIG. 1: Patterns of stabilizers of a distance $d = 3$ surface code with **a.** non-cyclic boundaries, **b.** cyclic boundaries. Examples of logical operators of each type have been given.

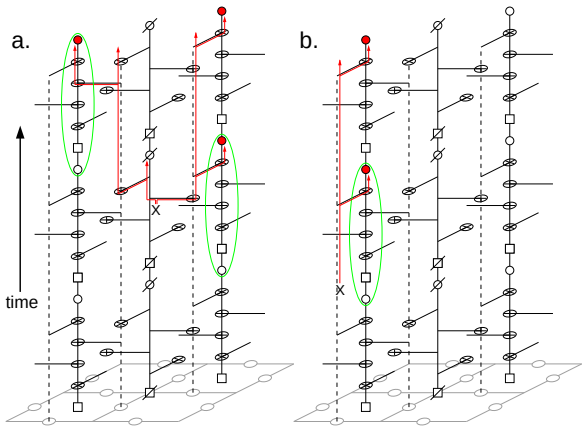

FIG. 2: 2-D surface code (grey). Time runs vertically. Squares represent initialization to $|0\rangle$, circles represent Z basis measurement. Slashed squares represent initialization to $|+\rangle$, slashed circles represent X basis measurement (both achieved using Hadamard gates). **a.** A single error leading to a pair of detection events (green ellipses). A detection event is a sequential pair of measurements with differing value. Red lines show the paths of error propagation. **b.** An error leading to a single detection event due to proximity to a boundary of the lattice.

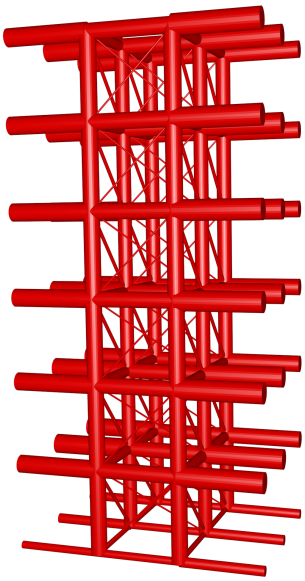

FIG. 3: Nest of 7 rounds of Fig. 1a fault-tolerant X stabilizer measurements.

in Figs. 5–6 using the work of [19]. These analytic expressions are shown as dashed lines and detailed in Table I and were derived completely independently of the simulations. The agreement with the simulation data is perfect to numerical precision, implying the processes leading to failure in the non-cyclic surface code are well understood.

As the non-cyclic code distance d increases, the number of minimum stick topologically nontrivial paths connecting opposing boundaries grows exponentially (consider larger versions of Fig. 3). Such paths can be associated with logical operators of minimum weight d, each of which crudely speaking increases the probability of logical error. More precise discussion can be found in [19]. By contrast, in a periodic, fault-tolerant, distance d surface code with cyclic boundaries, there are precisely d minimum stick topologically nontrivial cyclic paths. Diagonal sticks never zigzag and hence never create additional cyclic paths. This is a generic property of periodic, fault-tolerant TQEC. Logical errors will occur 50% of the time when physical errors occur corresponding to half the sticks in a given minimum weight logical operator. This implies the low p asymptotic logical error rate, for even

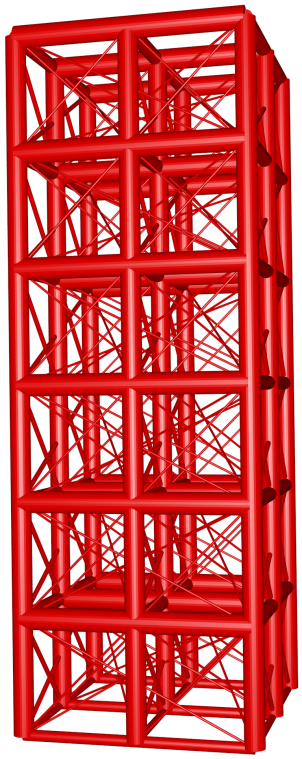

FIG. 4: Nest of 7 rounds of Fig. 1b fault-tolerant X stabilizer measurements.

d	A_X	A_Z	A_{X_1}	A_{Z_1}
4	3.97×10^2	4.70×10^2	1.23×10^2	2.76×10^2
6	1.67×10^4	2.09×10^4	1.97×10^3	6.64×10^3
8	7.02×10^5	9.34×10^5	2.94×10^4	1.49×10^5
10	2.93×10^7	4.18×10^7	4.23×10^5	3.21×10^6

TABLE I: Low p analytic asymptotic formulae derived from first principles, not simulation. Each entry represents a curve of the form $Ap^{d/2}$. The definitions of the logical operators X, Z, X_1, Z_1 can be found in Fig. 1.

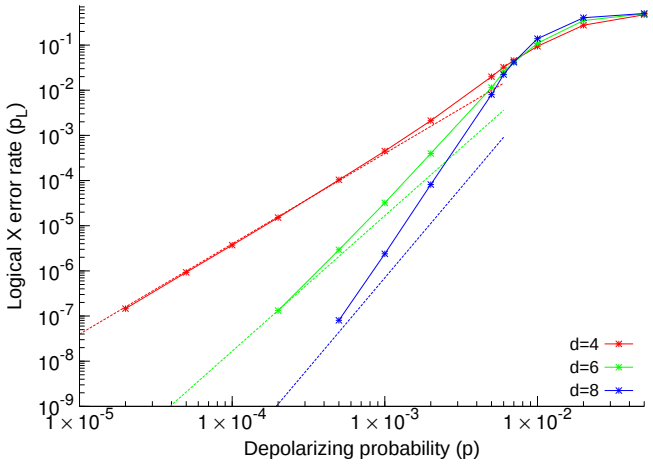

FIG. 5: Probability of logical X error as defined in Fig. 1a as a function of the depolarizing error rate p for various distances d. The analytic forms of the dashed curves can be found in Table I.

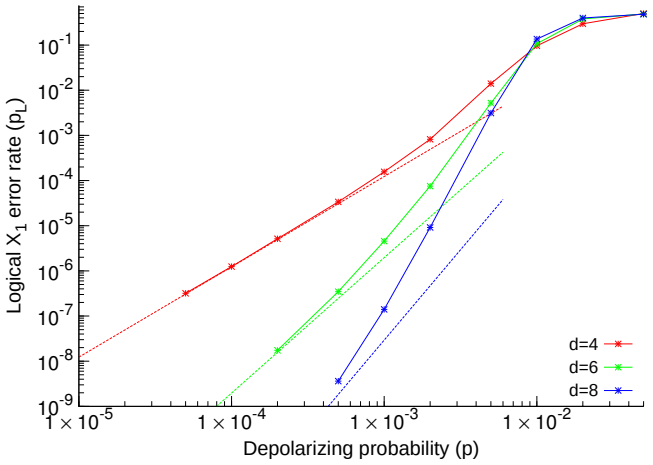

FIG. 7: Probability of logical X_1 error as defined in Fig. 1b as a function of the depolarizing error rate p for various distances d. The analytic forms of the dashed curves can be found in Table I.

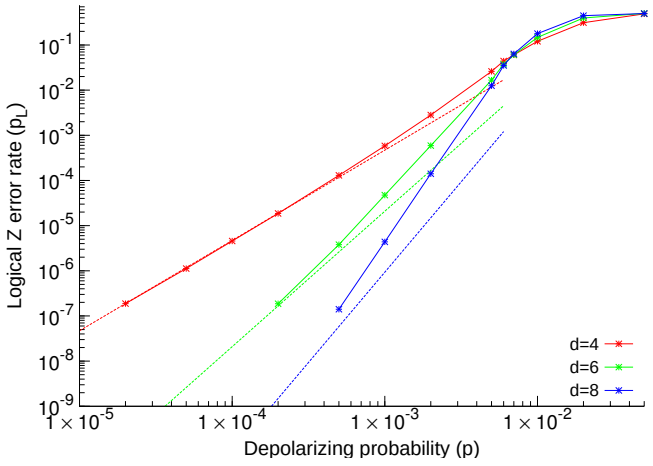

FIG. 6: Probability of logical Z error as defined in Fig. 1a as a function of the depolarizing error rate p for various distances d. The analytic forms of the dashed curves can be found in Table I.

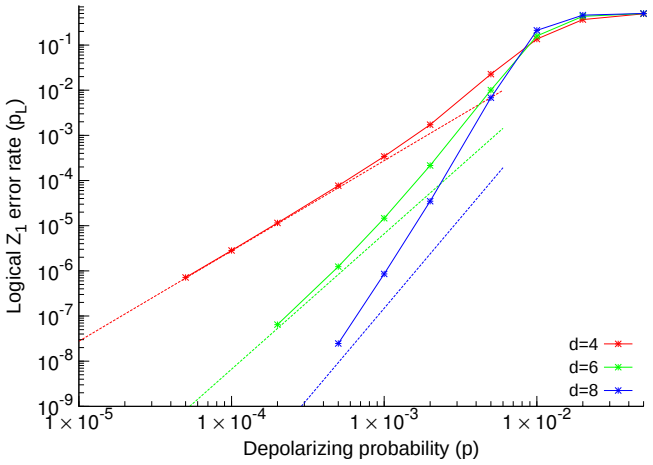

FIG. 8: Probability of logical Z_1 error as defined in Fig. 1b as a function of the depolarizing error rate p for various distances d. The analytic forms of the dashed curves can be found in Table I.

distances, will be

$$p_L = \frac{1}{2} d \begin{pmatrix} d \\ d/2 \end{pmatrix} \epsilon^{d/2}, \tag{1}$$

where ϵ is the probability of a stick within a minimum stick logical operator. Note that the symmetry of a cyclic boundary surface code implies that all such sticks will have the same probability.

From the Supplementary Material, we find that a Z_1 logical operator (left to right) is associated with sticks of probability $\epsilon = 4.8p$, for example by considering the $(4, 1, 6)$ to $(4, 3, 6)$ stick. A Z_2 logical operator (top to bottom) is associated with sticks of probability $\epsilon = 3.2p$. Inserting these expressions into Eq. 1 and noting that by symmetry the logical X_1 and Z_2 error rates will be the same, as will the logical X_2 and Z_1 error rates, we can see in Table I that at distance $d = 10$ the cyclic logical Z error rate is already over a factor of 10 too low and the cyclic logical X error rate nearly 2 orders of magnitude too low. The degree of underestimation grows exponentially with d.

In summary, we have shown that one of the most commonly used simplifying assumptions in physics and

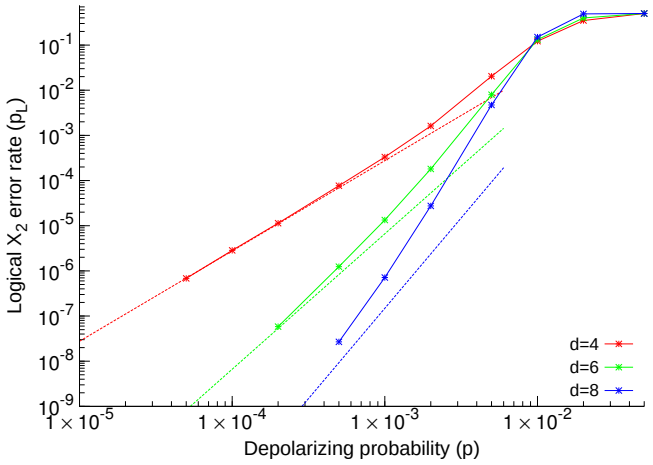

FIG. 9: Probability of logical X_2 error as defined in Fig. 1b as a function of the depolarizing error rate p for various distances d. The analytic forms of the dashed curves are identical to those of Fig. 8 and can be found in Table I.

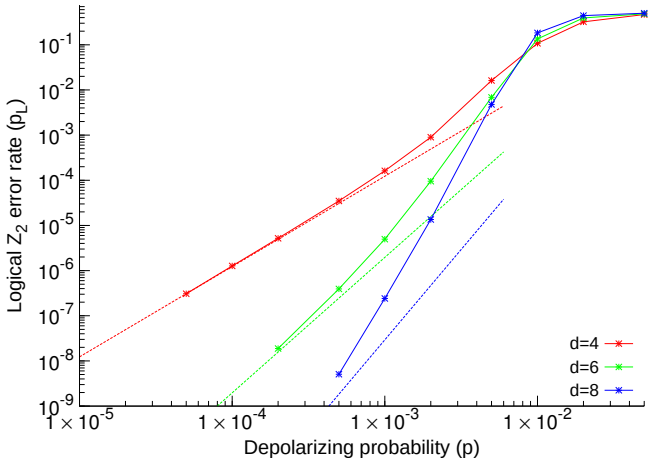

FIG. 10: Probability of logical Z_2 error as defined in Fig. 1b as a function of the depolarizing error rate p for various distances d. The analytic forms of the dashed curves are identical to those of Fig. 7 and can be found in Table I.

mathematics, namely cyclic boundaries, cannot be used if one wishes to accurately study the logical error rates of planar topological codes. The level of inaccuracy introduced by this seemingly harmless assumption is exponentially large in the code distance d. This implies that cyclic boundaries should be avoided when performing accurate studies of practical topological quantum error correction. We hope that this work encourages the quantum information community to seriously question the accuracy of results derived using unphysical assumptions. Two common assumptions that we feel deserve particular scrutiny are assuming perfect multi-body quantum measurements, and assuming arbitrarily long range coherent interactions without time, overhead, or error rate penalty.

I. ACKNOWLEDGEMENTS

This research was conducted by the Australian Research Council Centre of Excellence for Quantum Computation and Communication Technology (project number CE110001027). Supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center contract number D11PC20166. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

[1] A. G. Fowler and S. J. Devitt, arXiv:1209.0510 (2012).
[2] S. D. Barrett and T. M. Stace, Phys. Rev. Lett. **105**, 200502 (2010), arXiv:1005.2456.
[3] S. Bravyi and J. Haah, arXiv:1112.3252 (2011).
[4] S. Bravyi, G. Duclos-Cianci, D. Poulin, and M. Suchara, arXiv:1207.1443 (2012).
[5] R. S. Andrist, H. Bombin, H. G. Katzgraber, and M. A. Martin-Delgado, Phys. Rev. A **85**, 050302R (2012), arXiv:1204.1838.
[6] S. B. Bravyi and A. Y. Kitaev, quant-ph/9811052 (1998).
[7] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, J. Math. Phys. **43**, 4452 (2002), quant-ph/0110143.
[8] R. Raussendorf and J. Harrington, Phys. Rev. Lett. **98**, 190504 (2007), quant-ph/0610082.
[9] R. Raussendorf, J. Harrington, and K. Goyal, New J. Phys. **9**, 199 (2007), quant-ph/0703143.
[10] A. G. Fowler, A. M. Stephens, and P. Groszkowski, Phys. Rev. A **80**, 052312 (2009), arXiv:0803.0272.
[11] J. Edmonds, Canad. J. Math. **17**, 449 (1965).
[12] J. Edmonds, J. Res. Nat. Bur. Standards **69B**, 125

(1965).
[13] A. G. Fowler, A. C. Whiteside, and L. C. L. Hollenberg, Phys. Rev. Lett. **108**, 180501 (2012), arXiv:1110.5133.
[14] A. G. Fowler, A. C. Whiteside, and L. C. L. Hollenberg, arXiv:1202.5602 (2012).
[15] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, arXiv:1208.0928 (2012).
[16] D. Gottesman, Ph.D. thesis, Caltech (1997), quant-ph/9705052.
[17] A. G. Fowler, A. C. Whiteside, A. L. McInnes, and A. Rabbani, arXiv:1202.6111 (2012).
[18] D. S. Wang, A. G. Fowler, and L. C. L. Hollenberg, Phys. Rev. A **83**, 020302(R) (2011), arXiv:1009.3686.
[19] A. G. Fowler, arXiv:1208.1334 (2012).

Appendix A: Supplementary Material

We include below the raw data for the nests of Fig. 3 and Fig. 4. The initial two groups of three numbers in each line are the space-time coordinates of a cylinder, with the i, j coordinates corresponding to the numbers in Fig. 1, and the t coordinate corresponding to the number of rounds of error detection since the first, doubled. The time t therefore takes the values $\{0, 2, \ldots, 12\}$. The doubling of the time coordinate is purely to improve the symmetry of the nests. The final number in each line is the diameter of each cylinder and is the total probability of detection events at the end points of each cylinder from single errors when using a standard balanced depolarizing error model of strength $p = 0.04$. This value of p was chosen simply to produce a clear nest. The diameter of each cylinder scales linearly with p.

1. Data for Fig. 3

```
[
 [(4, 3, 12), (4, 3, 10), 0.1920499],
 [(4, 3, 12), (4, 1, 10), 0.0213650],
 [(4, 3, 12), (4, 5, 12), 0.2402681],
 [(4, 3, 12), (2, 3, 10), 0.0426811],
 [(4, 1, 12), (4, 3, 12), 0.1867471],
 [(4, 1, 12), (4, 1, 10), 0.1920499],
 [(4, 1, 12), (4, -1, 12), 0.2135253],
 [(4, 1, 12), (2, 3, 10), 0.0213650],
 [(4, 1, 12), (2, 1, 10), 0.0426811],
 [(2, 3, 12), (4, 3, 12), 0.1282210],
 [(2, 3, 12), (2, 3, 10), 0.1762239],
 [(2, 3, 12), (2, 1, 10), 0.0213650],
 [(2, 3, 12), (2, 5, 12), 0.2456125],
 [(2, 3, 12), (0, 3, 10), 0.0426811],
 [(2, 1, 12), (4, 1, 12), 0.1282210],
 [(2, 1, 12), (2, 3, 12), 0.1920499],
 [(2, 1, 12), (2, 1, 10), 0.1762239],
 [(2, 1, 12), (2, -1, 12), 0.2456125],
 [(2, 1, 12), (0, 3, 10), 0.0213650],
 [(2, 1, 12), (0, 1, 10), 0.0426811],
 [(0, 3, 12), (2, 3, 12), 0.1282210],
 [(0, 3, 12), (0, 3, 10), 0.1920499],
 [(0, 3, 12), (0, 1, 10), 0.0213650],
 [(0, 3, 12), (0, 5, 12), 0.2135253],
 [(0, 1, 12), (2, 1, 12), 0.1282210],
 [(0, 1, 12), (0, 3, 12), 0.1867471],
 [(0, 1, 12), (0, 1, 10), 0.1920499],
 [(0, 1, 12), (0, -1, 12), 0.2402681],
 [(4, 3, 10), (4, 3, 8), 0.1920499],
 [(4, 3, 10), (4, 1, 8), 0.0213650],
 [(4, 3, 10), (4, 5, 10), 0.2402681],
 [(4, 3, 10), (2, 3, 8), 0.0426811],
 [(4, 1, 10), (4, 3, 10), 0.1867471],
 [(4, 1, 10), (4, 1, 8), 0.1920499],
 [(4, 1, 10), (4, -1, 10), 0.2135253],
 [(4, 1, 10), (2, 3, 8), 0.0213650],
 [(4, 1, 10), (2, 1, 8), 0.0426811],
 [(2, 3, 10), (4, 3, 10), 0.1282210],
 [(2, 3, 10), (2, 3, 8), 0.1762239],
 [(2, 3, 10), (2, 1, 8), 0.0213650],
 [(2, 3, 10), (2, 5, 10), 0.2456125],
 [(2, 3, 10), (0, 3, 8), 0.0426811],
 [(2, 1, 10), (4, 1, 10), 0.1282210],
 [(2, 1, 10), (2, 3, 10), 0.1920499],
 [(2, 1, 10), (2, 1, 8), 0.1762239],
 [(2, 1, 10), (2, -1, 10), 0.2456125],
 [(2, 1, 10), (0, 3, 8), 0.0213650],
 [(2, 1, 10), (0, 1, 8), 0.0426811],
 [(0, 3, 10), (2, 3, 10), 0.1282210],
 [(0, 3, 10), (0, 3, 8), 0.1920499],
 [(0, 3, 10), (0, 1, 8), 0.0213650],
 [(0, 3, 10), (0, 5, 10), 0.2135253],
 [(0, 1, 10), (2, 1, 10), 0.1282210],
 [(0, 1, 10), (0, 3, 10), 0.1867471],
 [(0, 1, 10), (0, 1, 8), 0.1920499],
 [(0, 1, 10), (0, -1, 10), 0.2402681],
 [(4, 3, 8), (4, 3, 6), 0.1920499],
 [(4, 3, 8), (4, 1, 6), 0.0213650],
 [(4, 3, 8), (4, 5, 8), 0.2402681],
 [(4, 3, 8), (2, 3, 6), 0.0426811],
 [(4, 1, 8), (4, 3, 8), 0.1867471],
 [(4, 1, 8), (4, 1, 6), 0.1920499],
 [(4, 1, 8), (4, -1, 8), 0.2135253],
 [(4, 1, 8), (2, 3, 6), 0.0213650],
 [(4, 1, 8), (2, 1, 6), 0.0426811],
 [(2, 3, 8), (4, 3, 8), 0.1282210],
 [(2, 3, 8), (2, 3, 6), 0.1762239],
 [(2, 3, 8), (2, 1, 6), 0.0213650],
 [(2, 3, 8), (2, 5, 8), 0.2456125],
 [(2, 3, 8), (0, 3, 6), 0.0426811],
 [(2, 1, 8), (4, 1, 8), 0.1282210],
 [(2, 1, 8), (2, 3, 8), 0.1920499],
 [(2, 1, 8), (2, 1, 6), 0.1762239],
 [(2, 1, 8), (2, -1, 8), 0.2456125],
 [(2, 1, 8), (0, 3, 6), 0.0213650],
 [(2, 1, 8), (0, 1, 6), 0.0426811],
 [(0, 3, 8), (2, 3, 8), 0.1282210],
 [(0, 3, 8), (0, 3, 6), 0.1920499],
 [(0, 3, 8), (0, 1, 6), 0.0213650],
```

```
[(0, 3, 8), (0, 5, 8), 0.2135253],
[(0, 1, 8), (2, 1, 8), 0.1282210],
[(0, 1, 8), (0, 3, 8), 0.1867471],
[(0, 1, 8), (0, 1, 6), 0.1920499],
[(0, 1, 8), (0, -1, 8), 0.2402681],
[(4, 3, 6), (4, 3, 4), 0.1920499],
[(4, 3, 6), (4, 1, 4), 0.0213650],
[(4, 3, 6), (4, 5, 6), 0.2402681],
[(4, 3, 6), (2, 3, 4), 0.0426811],
[(4, 1, 6), (4, 3, 6), 0.1867471],
[(4, 1, 6), (4, 1, 4), 0.1920499],
[(4, 1, 6), (4, -1, 6), 0.2135253],
[(4, 1, 6), (2, 3, 4), 0.0213650],
[(4, 1, 6), (2, 1, 4), 0.0426811],
[(2, 3, 6), (4, 3, 6), 0.1282210],
[(2, 3, 6), (2, 3, 4), 0.1762239],
[(2, 3, 6), (2, 1, 4), 0.0213650],
[(2, 3, 6), (2, 5, 6), 0.2456125],
[(2, 3, 6), (0, 3, 4), 0.0426811],
[(2, 1, 6), (4, 1, 6), 0.1282210],
[(2, 1, 6), (2, 3, 6), 0.1920499],
[(2, 1, 6), (2, 1, 4), 0.1762239],
[(2, 1, 6), (2, -1, 6), 0.2456125],
[(2, 1, 6), (0, 3, 4), 0.0213650],
[(2, 1, 6), (0, 1, 4), 0.0426811],
[(0, 3, 6), (2, 3, 6), 0.1282210],
[(0, 3, 6), (0, 3, 4), 0.1920499],
[(0, 3, 6), (0, 1, 4), 0.0213650],
[(0, 3, 6), (0, 5, 6), 0.2135253],
[(0, 1, 6), (2, 1, 6), 0.1282210],
[(0, 1, 6), (0, 3, 6), 0.1867471],
[(0, 1, 6), (0, 1, 4), 0.1920499],
[(0, 1, 6), (0, -1, 6), 0.2402681],
[(4, 3, 4), (4, 3, 2), 0.1920499],
[(4, 3, 4), (4, 1, 2), 0.0213650],
[(4, 3, 4), (4, 5, 4), 0.2402681],
[(4, 3, 4), (2, 3, 2), 0.0426811],
[(4, 1, 4), (4, 3, 4), 0.1867471],
[(4, 1, 4), (4, 1, 2), 0.1920499],
[(4, 1, 4), (4, -1, 4), 0.2135253],
[(4, 1, 4), (2, 3, 2), 0.0213650],
[(4, 1, 4), (2, 1, 2), 0.0426811],
[(2, 3, 4), (4, 3, 4), 0.1282210],
[(2, 3, 4), (2, 3, 2), 0.1762239],
[(2, 3, 4), (2, 1, 2), 0.0213650],
[(2, 3, 4), (2, 5, 4), 0.2456125],
[(2, 3, 4), (0, 3, 2), 0.0426811],
[(2, 1, 4), (4, 1, 4), 0.1282210],
[(2, 1, 4), (2, 3, 4), 0.1920499],
[(2, 1, 4), (2, 1, 2), 0.1762239],
[(2, 1, 4), (2, -1, 4), 0.2456125],
[(2, 1, 4), (0, 3, 2), 0.0213650],
[(2, 1, 4), (0, 1, 2), 0.0426811],
[(0, 3, 4), (2, 3, 4), 0.1282210],
[(0, 3, 4), (0, 3, 2), 0.1920499],
[(0, 3, 4), (0, 1, 2), 0.0213650],
[(0, 3, 4), (0, 5, 4), 0.2135253],
[(0, 1, 4), (2, 1, 4), 0.1282210],
[(0, 1, 4), (0, 3, 4), 0.1867471],
[(0, 1, 4), (0, 1, 2), 0.1920499],
[(0, 1, 4), (0, -1, 4), 0.2402681],
[(4, 3, 2), (4, 3, 0), 0.1920499],
[(4, 3, 2), (4, 1, 0), 0.0213650],
[(4, 3, 2), (4, 5, 2), 0.2402681],
[(4, 3, 2), (2, 3, 0), 0.0426811],
[(4, 1, 2), (4, 3, 2), 0.1867471],
[(4, 1, 2), (4, 1, 0), 0.1920499],
[(4, 1, 2), (4, -1, 2), 0.2135253],
[(4, 1, 2), (2, 3, 0), 0.0213650],
[(4, 1, 2), (2, 1, 0), 0.0426811],
[(2, 3, 2), (4, 3, 2), 0.1282210],
[(2, 3, 2), (2, 3, 0), 0.1762239],
[(2, 3, 2), (2, 1, 0), 0.0213650],
[(2, 3, 2), (2, 5, 2), 0.2456125],
[(2, 3, 2), (0, 3, 0), 0.0426811],
[(2, 1, 2), (4, 1, 2), 0.1282210],
[(2, 1, 2), (2, 3, 2), 0.1920499],
[(2, 1, 2), (2, 1, 0), 0.1762239],
[(2, 1, 2), (2, -1, 2), 0.2456125],
[(2, 1, 2), (0, 3, 0), 0.0213650],
[(2, 1, 2), (0, 1, 0), 0.0426811],
[(0, 3, 2), (2, 3, 2), 0.1282210],
[(0, 3, 2), (0, 3, 0), 0.1920499],
[(0, 3, 2), (0, 1, 0), 0.0213650],
[(0, 3, 2), (0, 5, 2), 0.2135253],
[(0, 1, 2), (2, 1, 2), 0.1282210],
[(0, 1, 2), (0, 3, 2), 0.1867471],
[(0, 1, 2), (0, 1, 0), 0.1920499],
[(0, 1, 2), (0, -1, 2), 0.2402681],
[(4, 3, 0), (4, 5, 0), 0.1174198],
[(4, 1, 0), (4, 3, 0), 0.0907179],
[(4, 1, 0), (4, -1, 0), 0.0907179],
[(2, 3, 0), (4, 3, 0), 0.0640559],
[(2, 3, 0), (2, 5, 0), 0.1227016],
[(2, 1, 0), (4, 1, 0), 0.0640559],
[(2, 1, 0), (2, 3, 0), 0.0961352],
[(2, 1, 0), (2, -1, 0), 0.1227016],
[(0, 3, 0), (2, 3, 0), 0.0640559],
[(0, 3, 0), (0, 5, 0), 0.1227016],
[(0, 1, 0), (2, 1, 0), 0.0640559],
[(0, 1, 0), (0, 3, 0), 0.0961352],
[(0, 1, 0), (0, -1, 0), 0.1227016]
]
```

2. Data for Fig. 4

```
[
 [(4, 5, 12), (4, 5, 10), 0.1760737],
 [(4, 5, 12), (4, 3, 10), 0.0213468],
 [(4, 5, 12), (2, 5, 10), 0.0427301],
 [(4, 5, 12), (2, 1, 10), 0.0213468],
 [(4, 3, 12), (4, 5, 12), 0.1922704],
 [(4, 3, 12), (4, 3, 10), 0.1760737],
 [(4, 3, 12), (4, 1, 10), 0.0213468],
 [(4, 3, 12), (2, 5, 10), 0.0213468],
```

```
[(4, 3, 12), (2, 3, 10), 0.0427301],
[(4, 1, 12), (4, 5, 12), 0.1922704],
[(4, 1, 12), (4, 5, 10), 0.0213468],
[(4, 1, 12), (4, 3, 12), 0.1922704],
[(4, 1, 12), (4, 1, 10), 0.1760737],
[(4, 1, 12), (2, 3, 10), 0.0213468],
[(4, 1, 12), (2, 1, 10), 0.0427301],
[(2, 5, 12), (4, 5, 12), 0.1281117],
[(2, 5, 12), (2, 5, 10), 0.1760737],
[(2, 5, 12), (2, 3, 10), 0.0213468],
[(2, 5, 12), (0, 5, 10), 0.0427301],
[(2, 5, 12), (0, 1, 10), 0.0213468],
[(2, 3, 12), (4, 3, 12), 0.1281117],
[(2, 3, 12), (2, 5, 12), 0.1922704],
[(2, 3, 12), (2, 3, 10), 0.1760737],
[(2, 3, 12), (2, 1, 10), 0.0213468],
[(2, 3, 12), (0, 5, 10), 0.0213468],
[(2, 3, 12), (0, 3, 10), 0.0427301],
[(2, 1, 12), (4, 1, 12), 0.1281117],
[(2, 1, 12), (2, 5, 12), 0.1922704],
[(2, 1, 12), (2, 5, 10), 0.0213468],
[(2, 1, 12), (2, 3, 12), 0.1922704],
[(2, 1, 12), (2, 1, 10), 0.1760737],
[(2, 1, 12), (0, 3, 10), 0.0213468],
[(2, 1, 12), (0, 1, 10), 0.0427301],
[(0, 5, 12), (4, 5, 12), 0.1281117],
[(0, 5, 12), (4, 5, 10), 0.0427301],
[(0, 5, 12), (4, 1, 10), 0.0213468],
[(0, 5, 12), (2, 5, 12), 0.1281117],
[(0, 5, 12), (0, 5, 10), 0.1760737],
[(0, 5, 12), (0, 3, 10), 0.0213468],
[(0, 3, 12), (4, 5, 10), 0.0213468],
[(0, 3, 12), (4, 3, 12), 0.1281117],
[(0, 3, 12), (4, 3, 10), 0.0427301],
[(0, 3, 12), (2, 3, 12), 0.1281117],
[(0, 3, 12), (0, 5, 12), 0.1922704],
[(0, 3, 12), (0, 3, 10), 0.1760737],
[(0, 3, 12), (0, 1, 10), 0.0213468],
[(0, 1, 12), (4, 3, 10), 0.0213468],
[(0, 1, 12), (4, 1, 12), 0.1281117],
[(0, 1, 12), (4, 1, 10), 0.0427301],
[(0, 1, 12), (2, 1, 12), 0.1281117],
[(0, 1, 12), (0, 5, 12), 0.1922704],
[(0, 1, 12), (0, 5, 10), 0.0213468],
[(0, 1, 12), (0, 3, 12), 0.1922704],
[(0, 1, 12), (0, 1, 10), 0.1760737],
[(4, 5, 10), (4, 5, 8), 0.1760737],
[(4, 5, 10), (4, 3, 8), 0.0213468],
[(4, 5, 10), (2, 5, 8), 0.0427301],
[(4, 5, 10), (2, 1, 8), 0.0213468],
[(4, 3, 10), (4, 5, 10), 0.1922704],
[(4, 3, 10), (4, 3, 8), 0.1760737],
[(4, 3, 10), (4, 1, 8), 0.0213468],
[(4, 3, 10), (2, 5, 8), 0.0213468],
[(4, 3, 10), (2, 3, 8), 0.0427301],
[(4, 1, 10), (4, 5, 10), 0.1922704],
[(4, 1, 10), (4, 5, 8), 0.0213468],
[(4, 1, 10), (4, 3, 10), 0.1922704],
[(4, 1, 10), (4, 1, 8), 0.1760737],
[(4, 1, 10), (2, 3, 8), 0.0213468],
[(4, 1, 10), (2, 1, 8), 0.0427301],
[(2, 5, 10), (4, 5, 10), 0.1281117],
[(2, 5, 10), (2, 5, 8), 0.1760737],
[(2, 5, 10), (2, 3, 8), 0.0213468],
[(2, 5, 10), (0, 5, 8), 0.0427301],
[(2, 5, 10), (0, 1, 8), 0.0213468],
[(2, 3, 10), (4, 3, 10), 0.1281117],
[(2, 3, 10), (2, 5, 10), 0.1922704],
[(2, 3, 10), (2, 3, 8), 0.1760737],
[(2, 3, 10), (2, 1, 8), 0.0213468],
[(2, 3, 10), (0, 5, 8), 0.0213468],
[(2, 3, 10), (0, 3, 8), 0.0427301],
[(2, 1, 10), (4, 1, 10), 0.1281117],
[(2, 1, 10), (2, 5, 10), 0.1922704],
[(2, 1, 10), (2, 5, 8), 0.0213468],
[(2, 1, 10), (2, 3, 10), 0.1922704],
[(2, 1, 10), (2, 1, 8), 0.1760737],
[(2, 1, 10), (0, 3, 8), 0.0213468],
[(2, 1, 10), (0, 1, 8), 0.0427301],
[(0, 5, 10), (4, 5, 10), 0.1281117],
[(0, 5, 10), (4, 5, 8), 0.0427301],
[(0, 5, 10), (4, 1, 8), 0.0213468],
[(0, 5, 10), (2, 5, 10), 0.1281117],
[(0, 5, 10), (0, 5, 8), 0.1760737],
[(0, 5, 10), (0, 3, 8), 0.0213468],
[(0, 3, 10), (4, 5, 8), 0.0213468],
[(0, 3, 10), (4, 3, 10), 0.1281117],
[(0, 3, 10), (4, 3, 8), 0.0427301],
[(0, 3, 10), (2, 3, 10), 0.1281117],
[(0, 3, 10), (0, 5, 10), 0.1922704],
[(0, 3, 10), (0, 3, 8), 0.1760737],
[(0, 3, 10), (0, 1, 8), 0.0213468],
[(0, 1, 10), (4, 3, 8), 0.0213468],
[(0, 1, 10), (4, 1, 10), 0.1281117],
[(0, 1, 10), (4, 1, 8), 0.0427301],
[(0, 1, 10), (2, 1, 10), 0.1281117],
[(0, 1, 10), (0, 5, 10), 0.1922704],
[(0, 1, 10), (0, 5, 8), 0.0213468],
[(0, 1, 10), (0, 3, 10), 0.1922704],
[(0, 1, 10), (0, 1, 8), 0.1760737],
[(4, 5, 8), (4, 5, 6), 0.1760737],
[(4, 5, 8), (4, 3, 6), 0.0213468],
[(4, 5, 8), (2, 5, 6), 0.0427301],
[(4, 5, 8), (2, 1, 6), 0.0213468],
[(4, 3, 8), (4, 5, 8), 0.1922704],
[(4, 3, 8), (4, 3, 6), 0.1760737],
[(4, 3, 8), (4, 1, 6), 0.0213468],
[(4, 3, 8), (2, 5, 6), 0.0213468],
[(4, 3, 8), (2, 3, 6), 0.0427301],
[(4, 1, 8), (4, 5, 8), 0.1922704],
[(4, 1, 8), (4, 5, 6), 0.0213468],
[(4, 1, 8), (4, 3, 8), 0.1922704],
[(4, 1, 8), (4, 1, 6), 0.1760737],
[(4, 1, 8), (2, 3, 6), 0.0213468],
[(4, 1, 8), (2, 1, 6), 0.0427301],
[(2, 5, 8), (4, 5, 8), 0.1281117],
```

```
[(2, 5, 8), (2, 5, 6), 0.1760737],
[(2, 5, 8), (2, 3, 6), 0.0213468],
[(2, 5, 8), (0, 5, 6), 0.0427301],
[(2, 5, 8), (0, 1, 6), 0.0213468],
[(2, 3, 8), (4, 3, 8), 0.1281117],
[(2, 3, 8), (2, 5, 8), 0.1922704],
[(2, 3, 8), (2, 3, 6), 0.1760737],
[(2, 3, 8), (2, 1, 6), 0.0213468],
[(2, 3, 8), (0, 5, 6), 0.0213468],
[(2, 3, 8), (0, 3, 6), 0.0427301],
[(2, 1, 8), (4, 1, 8), 0.1281117],
[(2, 1, 8), (2, 5, 8), 0.1922704],
[(2, 1, 8), (2, 5, 6), 0.0213468],
[(2, 1, 8), (2, 3, 8), 0.1922704],
[(2, 1, 8), (2, 1, 6), 0.1760737],
[(2, 1, 8), (0, 3, 6), 0.0213468],
[(2, 1, 8), (0, 1, 6), 0.0427301],
[(0, 5, 8), (4, 5, 8), 0.1281117],
[(0, 5, 8), (4, 5, 6), 0.0427301],
[(0, 5, 8), (4, 1, 6), 0.0213468],
[(0, 5, 8), (2, 5, 8), 0.1281117],
[(0, 5, 8), (0, 5, 6), 0.1760737],
[(0, 5, 8), (0, 3, 6), 0.0213468],
[(0, 3, 8), (4, 5, 6), 0.0213468],
[(0, 3, 8), (4, 3, 8), 0.1281117],
[(0, 3, 8), (4, 3, 6), 0.0427301],
[(0, 3, 8), (2, 3, 8), 0.1281117],
[(0, 3, 8), (0, 5, 8), 0.1922704],
[(0, 3, 8), (0, 3, 6), 0.1760737],
[(0, 3, 8), (0, 1, 6), 0.0213468],
[(0, 1, 8), (4, 3, 6), 0.0213468],
[(0, 1, 8), (4, 1, 8), 0.1281117],
[(0, 1, 8), (4, 1, 6), 0.0427301],
[(0, 1, 8), (2, 1, 8), 0.1281117],
[(0, 1, 8), (0, 5, 8), 0.1922704],
[(0, 1, 8), (0, 5, 6), 0.0213468],
[(0, 1, 8), (0, 3, 8), 0.1922704],
[(0, 1, 8), (0, 1, 6), 0.1760737],
[(4, 5, 6), (4, 5, 4), 0.1760737],
[(4, 5, 6), (4, 3, 4), 0.0213468],
[(4, 5, 6), (2, 5, 4), 0.0427301],
[(4, 5, 6), (2, 1, 4), 0.0213468],
[(4, 3, 6), (4, 5, 6), 0.1922704],
[(4, 3, 6), (4, 3, 4), 0.1760737],
[(4, 3, 6), (4, 1, 4), 0.0213468],
[(4, 3, 6), (2, 5, 4), 0.0213468],
[(4, 3, 6), (2, 3, 4), 0.0427301],
[(4, 1, 6), (4, 5, 6), 0.1922704],
[(4, 1, 6), (4, 5, 4), 0.0213468],
[(4, 1, 6), (4, 3, 6), 0.1922704],
[(4, 1, 6), (4, 1, 4), 0.1760737],
[(4, 1, 6), (2, 3, 4), 0.0213468],
[(4, 1, 6), (2, 1, 4), 0.0427301],
[(2, 5, 6), (4, 5, 6), 0.1281117],
[(2, 5, 6), (2, 5, 4), 0.1760737],
[(2, 5, 6), (2, 3, 4), 0.0213468],
[(2, 5, 6), (0, 5, 4), 0.0427301],
[(2, 5, 6), (0, 1, 4), 0.0213468],
[(2, 3, 6), (4, 3, 6), 0.1281117],
[(2, 3, 6), (2, 5, 6), 0.1922704],
[(2, 3, 6), (2, 3, 4), 0.1760737],
[(2, 3, 6), (2, 1, 4), 0.0213468],
[(2, 3, 6), (0, 5, 4), 0.0213468],
[(2, 3, 6), (0, 3, 4), 0.0427301],
[(2, 1, 6), (4, 1, 6), 0.1281117],
[(2, 1, 6), (2, 5, 6), 0.1922704],
[(2, 1, 6), (2, 5, 4), 0.0213468],
[(2, 1, 6), (2, 3, 6), 0.1922704],
[(2, 1, 6), (2, 1, 4), 0.1760737],
[(2, 1, 6), (0, 3, 4), 0.0213468],
[(2, 1, 6), (0, 1, 4), 0.0427301],
[(0, 5, 6), (4, 5, 6), 0.1281117],
[(0, 5, 6), (4, 5, 4), 0.0427301],
[(0, 5, 6), (4, 1, 4), 0.0213468],
[(0, 5, 6), (2, 5, 6), 0.1281117],
[(0, 5, 6), (0, 5, 4), 0.1760737],
[(0, 5, 6), (0, 3, 4), 0.0213468],
[(0, 3, 6), (4, 5, 4), 0.0213468],
[(0, 3, 6), (4, 3, 6), 0.1281117],
[(0, 3, 6), (4, 3, 4), 0.0427301],
[(0, 3, 6), (2, 3, 6), 0.1281117],
[(0, 3, 6), (0, 5, 6), 0.1922704],
[(0, 3, 6), (0, 3, 4), 0.1760737],
[(0, 3, 6), (0, 1, 4), 0.0213468],
[(0, 1, 6), (4, 3, 4), 0.0213468],
[(0, 1, 6), (4, 1, 6), 0.1281117],
[(0, 1, 6), (4, 1, 4), 0.0427301],
[(0, 1, 6), (2, 1, 6), 0.1281117],
[(0, 1, 6), (0, 5, 6), 0.1922704],
[(0, 1, 6), (0, 5, 4), 0.0213468],
[(0, 1, 6), (0, 3, 6), 0.1922704],
[(0, 1, 6), (0, 1, 4), 0.1760737],
[(4, 5, 4), (4, 5, 2), 0.1760737],
[(4, 5, 4), (4, 3, 2), 0.0213468],
[(4, 5, 4), (2, 5, 2), 0.0427301],
[(4, 5, 4), (2, 1, 2), 0.0213468],
[(4, 3, 4), (4, 5, 4), 0.1922704],
[(4, 3, 4), (4, 3, 2), 0.1760737],
[(4, 3, 4), (4, 1, 2), 0.0213468],
[(4, 3, 4), (2, 5, 2), 0.0213468],
[(4, 3, 4), (2, 3, 2), 0.0427301],
[(4, 1, 4), (4, 5, 4), 0.1922704],
[(4, 1, 4), (4, 5, 2), 0.0213468],
[(4, 1, 4), (4, 3, 4), 0.1922704],
[(4, 1, 4), (4, 1, 2), 0.1760737],
[(4, 1, 4), (2, 3, 2), 0.0213468],
[(4, 1, 4), (2, 1, 2), 0.0427301],
[(2, 5, 4), (4, 5, 4), 0.1281117],
[(2, 5, 4), (2, 5, 2), 0.1760737],
[(2, 5, 4), (2, 3, 2), 0.0213468],
[(2, 5, 4), (0, 5, 2), 0.0427301],
[(2, 5, 4), (0, 1, 2), 0.0213468],
[(2, 3, 4), (4, 3, 4), 0.1281117],
[(2, 3, 4), (2, 5, 4), 0.1922704],
[(2, 3, 4), (2, 3, 2), 0.1760737],
[(2, 3, 4), (2, 1, 2), 0.0213468],
```

```
[(2, 3, 4), (0, 5, 2), 0.0213468],
[(2, 3, 4), (0, 3, 2), 0.0427301],
[(2, 1, 4), (4, 1, 4), 0.1281117],
[(2, 1, 4), (2, 5, 4), 0.1922704],
[(2, 1, 4), (2, 5, 2), 0.0213468],
[(2, 1, 4), (2, 3, 4), 0.1922704],
[(2, 1, 4), (2, 1, 2), 0.1760737],
[(2, 1, 4), (0, 3, 2), 0.0213468],
[(2, 1, 4), (0, 1, 2), 0.0427301],
[(0, 5, 4), (4, 5, 4), 0.1281117],
[(0, 5, 4), (4, 5, 2), 0.0427301],
[(0, 5, 4), (4, 1, 2), 0.0213468],
[(0, 5, 4), (2, 5, 4), 0.1281117],
[(0, 5, 4), (0, 5, 2), 0.1760737],
[(0, 5, 4), (0, 3, 2), 0.0213468],
[(0, 3, 4), (4, 5, 2), 0.0213468],
[(0, 3, 4), (4, 3, 4), 0.1281117],
[(0, 3, 4), (4, 3, 2), 0.0427301],
[(0, 3, 4), (2, 3, 4), 0.1281117],
[(0, 3, 4), (0, 5, 4), 0.1922704],
[(0, 3, 4), (0, 3, 2), 0.1760737],
[(0, 3, 4), (0, 1, 2), 0.0213468],
[(0, 1, 4), (4, 3, 2), 0.0213468],
[(0, 1, 4), (4, 1, 4), 0.1281117],
[(0, 1, 4), (4, 1, 2), 0.0427301],
[(0, 1, 4), (2, 1, 4), 0.1281117],
[(0, 1, 4), (0, 5, 4), 0.1922704],
[(0, 1, 4), (0, 5, 2), 0.0213468],
[(0, 1, 4), (0, 3, 4), 0.1922704],
[(0, 1, 4), (0, 1, 2), 0.1760737],
[(4, 5, 2), (4, 5, 0), 0.1760737],
[(4, 5, 2), (4, 3, 0), 0.0213468],
[(4, 5, 2), (2, 5, 0), 0.0427301],
[(4, 5, 2), (2, 1, 0), 0.0213468],
[(4, 3, 2), (4, 5, 2), 0.1922704],
[(4, 3, 2), (4, 3, 0), 0.1760737],
[(4, 3, 2), (4, 1, 0), 0.0213468],
[(4, 3, 2), (2, 5, 0), 0.0213468],
[(4, 3, 2), (2, 3, 0), 0.0427301],
[(4, 1, 2), (4, 5, 2), 0.1922704],
[(4, 1, 2), (4, 5, 0), 0.0213468],
[(4, 1, 2), (4, 3, 2), 0.1922704],
[(4, 1, 2), (4, 1, 0), 0.1760737],
[(4, 1, 2), (2, 3, 0), 0.0213468],
[(4, 1, 2), (2, 1, 0), 0.0427301],
[(2, 5, 2), (4, 5, 2), 0.1281117],
[(2, 5, 2), (2, 5, 0), 0.1760737],
[(2, 5, 2), (2, 3, 0), 0.0213468],
[(2, 5, 2), (0, 5, 0), 0.0427301],
[(2, 5, 2), (0, 1, 0), 0.0213468],
[(2, 3, 2), (4, 3, 2), 0.1281117],
[(2, 3, 2), (2, 5, 2), 0.1922704],
[(2, 3, 2), (2, 3, 0), 0.1760737],
[(2, 3, 2), (2, 1, 0), 0.0213468],
[(2, 3, 2), (0, 5, 0), 0.0213468],
[(2, 3, 2), (0, 3, 0), 0.0427301],
[(2, 1, 2), (4, 1, 2), 0.1281117],
[(2, 1, 2), (2, 5, 2), 0.1922704],
[(2, 1, 2), (2, 5, 0), 0.0213468],
[(2, 1, 2), (2, 3, 2), 0.1922704],
[(2, 1, 2), (2, 1, 0), 0.1760737],
[(2, 1, 2), (0, 3, 0), 0.0213468],
[(2, 1, 2), (0, 1, 0), 0.0427301],
[(0, 5, 2), (4, 5, 2), 0.1281117],
[(0, 5, 2), (4, 5, 0), 0.0427301],
[(0, 5, 2), (4, 1, 0), 0.0213468],
[(0, 5, 2), (2, 5, 2), 0.1281117],
[(0, 5, 2), (0, 5, 0), 0.1760737],
[(0, 5, 2), (0, 3, 0), 0.0213468],
[(0, 3, 2), (4, 5, 0), 0.0213468],
[(0, 3, 2), (4, 3, 2), 0.1281117],
[(0, 3, 2), (4, 3, 0), 0.0427301],
[(0, 3, 2), (2, 3, 2), 0.1281117],
[(0, 3, 2), (0, 5, 2), 0.1922704],
[(0, 3, 2), (0, 3, 0), 0.1760737],
[(0, 3, 2), (0, 1, 0), 0.0213468],
[(0, 1, 2), (4, 3, 0), 0.0213468],
[(0, 1, 2), (4, 1, 2), 0.1281117],
[(0, 1, 2), (4, 1, 0), 0.0427301],
[(0, 1, 2), (2, 1, 2), 0.1281117],
[(0, 1, 2), (0, 5, 2), 0.1922704],
[(0, 1, 2), (0, 5, 0), 0.0213468],
[(0, 1, 2), (0, 3, 2), 0.1922704],
[(0, 1, 2), (0, 1, 0), 0.1760737],
[(4, 3, 0), (4, 5, 0), 0.0960532],
[(4, 1, 0), (4, 5, 0), 0.0960532],
[(4, 1, 0), (4, 3, 0), 0.0960532],
[(2, 5, 0), (4, 5, 0), 0.0640012],
[(2, 3, 0), (4, 3, 0), 0.0640012],
[(2, 3, 0), (2, 5, 0), 0.0960532],
[(2, 1, 0), (4, 1, 0), 0.0640012],
[(2, 1, 0), (2, 5, 0), 0.0960532],
[(2, 1, 0), (2, 3, 0), 0.0960532],
[(0, 5, 0), (4, 5, 0), 0.0640012],
[(0, 5, 0), (2, 5, 0), 0.0640012],
[(0, 3, 0), (4, 3, 0), 0.0640012],
[(0, 3, 0), (2, 3, 0), 0.0640012],
[(0, 3, 0), (0, 5, 0), 0.0960532],
[(0, 1, 0), (4, 1, 0), 0.0640012],
[(0, 1, 0), (2, 1, 0), 0.0640012],
[(0, 1, 0), (0, 5, 0), 0.0960532],
[(0, 1, 0), (0, 3, 0), 0.0960532]
]
```